System Utilization and Changes in Implemented Information Systems: A Case Study

H. Keith Edwards, *Member, IAENG*

***Abstract*— The implementation phase of the software development life cycle is critical for continued organizational success. As organizations evolve their business processes after the launch of a new system, the system needs to continue to evolve in order to take advantage of these opportunities. In this research, we explore the changes that take place in a large manufacturing system in order to understand the relationships between system usage, business process change, and system change. Our research shows close relationships between the most utilized screens in the system, the core business process, and the changes requested by users of the system. We also examine the effect of prioritizing changes on the time to implement these changes.**

***Index Terms*— Configuration Management, Implementation Issues. Software Engineering, System Maintenance.**

I. Introduction

Proper management and control of the post implementation phase of the software engineering life cycle is critical for continued project success. While it is a well recognized fact that continued change is necessary in order for a system to remain useful over its lifetime, there is a paucity of case studies that examine the nature of post implementation project changes.

This paper examines the post-implementation phase of the software product life cycle for an inventory requisitioning and control system from a large manufacturing company in order to better understand the nature of system usage and system change requests. In particular, this paper explores how systems are used and the relationship between system usage, change requests, and business processes. Finally, this paper looks at the accuracy of systems estimates based on the priority assigned to the change request.

The rest of this paper is organized as follows. Section 2 examines work related to the areas of implemented information systems, configuration management strategies and support, and discusses the information provided by existing case studies. Section 3 provides a description of the implemented system and its environment, the change control process used by the company, and the data set. This section also examines the methods used for our analysis. Section 4 examines the results of our analysis. Section 5 provides conclusions and explores the applicability of these results to other implemented systems.

II. Related Work

Implementation and program maintenance is the final stage of the software development life cycle [16,30]. Sommerville [30] defines the implementation stage of the software development life cycle as "the process of converting a system specification into an executable system." The word process in this definition shows that this is not merely a destination, but an ongoing process that involves the continual modification of the system to meet the requirements of its operating environment.

The need for applying changes to implemented software has been evident since the development of the first systems. Early research in software engineering shows the need for applying a process in order to control changes that are incorporated into the final product [1,3]. Later work such as Joeris [14] looks at how to provide basic management function for both change management and configuration management within the implementation phase of the software development life cycle, while Davis examines the role that the overall development life cycle plays in the software configuration management process in the post waterfall model era [6].

An underlying consideration to any view of the change management process is that large scale information systems constitute socio-technical systems that interleave people, process, and technology [30]. Leavitt's model (Figure 1) is perhaps the seminal piece of related work in the arena of management information systems for understanding the role of the implemented system within the overall organization [17]. Leavitt's model examines the relationship between business process, change, management processes, and information systems. The model hypothesizes that changes to any one aspect of the model will cause changes in the other areas of the model so that the organization can maintain its overall strategic alignment. Since this research examines changes to the system resulting primarily from changes to the business process (as opposed to pure technological changes), we can employ Leavitt's model as a tool for understanding the relationships between several of the aspects of the implemented system.

H. Keith Edwards is with the University of Hawaii at Hilo, 200 W. Kawili Street Hilo, HI 96720 USA (phone: 808-933-3189; fax: (808) 974-7693; e-mail: hedwards@hawaii.edu).

(Advance online publication: 13 February 2007)

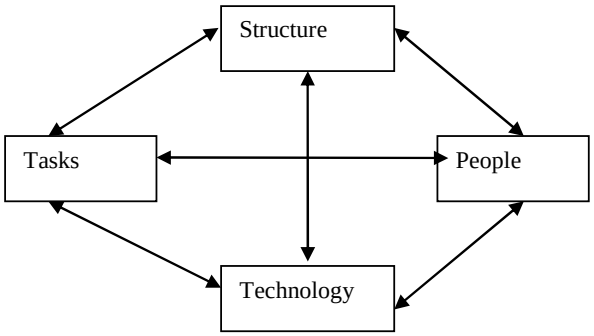

FIGURE 1 – LEAVITT'S MODEL

A. Change Control Process and Configuration Management Strategies

Several articles in the related literature point to the need for and examine the nature of change control and configuration management processes that form a critical part of the implementation phase of the software development life cycle [2,32]. For example, Joeris [14] provides a summary of desirable characteristics for a change management process such as managing the process of change as well as managing the artifacts of the change process. Chillakanti [5] motivates the need for incorporating security into the change management process. Sato, et al. [24] discuss Hewlett Packard's change management process and its impact on the customer experience, but focuses on off the shelf products as opposed to bespoke systems. Nguyen [21,22,23] examines the software configuration environment in object oriented and web-based systems, but treats the more general subject matter than a specific system. Other work such as [26] looks at configuration control for evolutionary systems, whilst Simmonds [27] examines the configuration management process on the PACT software engineering system.

While the need for a disciplined change management process can not be understated, the purpose of this research is not to supplant any of these existing processes nor to examine the particular configuration of the software from a technical standpoint. Rather, this research examines the process only as it relates to the implementation of the changes within the system and do not treat it from a theoretical standpoint. Further to this end, we accept the process as an invariant within the overall business environment.

B. Support for Configuration Management e

Calabough [4] points to the need for change and configuration management support and develops a tool along with a process to these ends. While similar in its discussion of the process, it differs in that there is no discussion of the changes that were actually incorporated into the system.

Several articles [12,18,19,25,31] examine the configuration management process in order to manage component versions in the build. While these tools act as a support for the continuing evolution of software products, they deal more with managing the technical aspects of change than with the business process. Feiler [11] examines how process support can assist in the implementation of configuration management tools. A general treatment of support for configuration management can be found in Estublier [8,9,10] and in German, et al. [13], who provides a framework for describing tools for the mining of software repositories.

C. Configuration Management Case Studies

A final area of research is that of configuration management case studies. These are sparse, but will most likely differ from our research in the scale of the effort (toy systems), the amount of data examined, and the time over which the system was examined. Our study extends for five years and examines approximately 600 separate change requests.

Dietel [7] provides a look at the impact of instituting a change management process into the organization and how this impacts the employees in the organization.

Sliwerski, et al. provides a study of changes in open source software such as Mozilla and Eclipse [28]. Their research examined the impact of fix inducing changes on the system and provided recommendations as to how to minimize later fixes due to these changes. Further work by this group [29] looks at how to relate the software version history to the bug database.

Likewise, [20] uses data on Mozilla and Apache to understand source code change history, problem reports, and productivity in order to understand the open source development process. Finally, Koponen [15] looks at the open source maintenance process for these same two products.

While all of these case studies examine the post implementation phase of large software systems, they deal with open-source off the shelf products designed for the mass market as opposed to bespoke systems that are designed to provide a competitive advantage to a business in a proprietary manner. The nature of the systems under consideration as well as the lack of focus on business process in these case studies clearly differentiates our work from these studies.

III. DESCRIPTION OF SYSTEM ENVIRONMENT, CHANGE CONTROL PROCESS, DATA SET, AND METHODS FOR ANALYSIS

In this section of the paper, we describe the particulars of our research. In particular, we describe the system under study, the environment in which it operates as well as the configuration management process for managing changes to this particular information system and the data recorded as part of that process. Finally, we discuss our method for extracting and analyzing the data and the questions that we seek to answer as part and parcel to this investigation.

A. System and Environment

In this research, we examine the post-implementation history

of an inventory and requisitioning system used by a large global manufacturer. The overall system itself is divided into four main modules that represent the core business processes of the company. The first module is responsible for the specification and scheduling of the individual products. The second module allows users to specify a bill of material for individual products, i.e. to indicate which individual parts are used in a particular product. The third module supports the ordering of the parts from the bill of material, and the final module supports warehousing and logistic operations such as receiving, storing, allocating, and shipping the individual parts.

The system was originally released in the United States, United Kingdom, and Germany in 1997. It is now deployed in seven different countries and has thousands of individual users. It currently has 325 different screens that support queries, updates, and administrative functions.

The system is supported by full time staff who are dedicated to representing the users and who are also responsible for explaining and cascading the business process throughout the organization. There is also a systems staff that is responsible for ensuring that the system functions correctly from a technical standpoint. These two groups work together in order to facilitate a holistic approach to system configuration and maintenance.

B. Configuration Management Process

Since this is a bespoke system that was designed to support the business process, it is not surprising that the system would need to change as the business process continued to evolve. The configuration management process is designed to support changes that are required in order to support the business process. As such, there is a separate process designated to deal with fixes to the system that are a result of bugs in the software.

The change control process begins with an individual user requesting a change to the system. This change is analyzed by a business process analyst who determines whether the change is needed to support the business process. If the change is required to support the business process, then the analyst assigns a priority to the change. The business process analyst then meets with representatives from the systems staff who analyze the request to see if it is actually required and if it is technically feasible. If both of these conditions are met, the systems group negotiates a target date with the business process group and provides an estimated cost for the change.

The system group then implements the change in the development system and reports the completion of the change to the business process analysts. The change is then tested and if it results in no errors, it is released into the production environment of the system as part of a batch. Batches are released into the production system several times throughout the year. Finally, information about the changes is cascaded to other support areas such as training and the help desk.

C. Description of the Data Set

For the purposes of this study, we looked at two artifacts of the post implementation phase of the system. First, we examined a synopsis of the change control requests for the system. This synopsis provided summary information about the 585 change controls that are currently in the system. In particular, this archive provides the following information:

- Change Request Number
- System Area
- A description of the change request
- The date of the change request
- The status of the change request (new, evaluation, accepted, implemented, canceled, rejected, hold) and the status date
- The priority of the change request
- The system hours for the request and the expected benefit to the company of the request
- The target date for the systems group
- The actual completion date
- Check boxes for updating of the business process, training, and help screens.

The second document in the data set is the access information for individual screens in the system. This access information is broken down by year, beginning in 2000 and running until the year 2005. It indicates the screen number, screen name, module, and the number of "hits" for that screen for each of the years that the screen was active. Several low use screens were phased out during this time and only contained data for 1-2 year.

D. Research Questions

In this research, we sought to quantitatively answer several questions about the post implementation operation of the information system. In particular, we wanted to examine the following aspects of the system:

- Screen usage patterns and their relationship to change requests.
- The prioritization of changes and the accuracy of systems estimates based on priority of changes.

IV. Analysis and Results

To gain answers to the questions in the previous, we calculated the average of screen usage patterns for the entire six year period. We placed this screens in order of their average number of accesses per year and examined the use of the top 10% and top quartile of the screens to determine the percentage of the overall traffic that they receive. Our results, shown in Table 1, indicate an empirical validation of the "80-20" rule.

Total Screen Use	17332792	
Top 10% Screen Use	13770566	79.45
Top 25% Screen Use	16720453	96.47

Table 1 – Hits for the most used screens and by module

Here, we can see that the top quartile (25%) of the screens receive about 96% of the traffic while the top 10% of the screens receive 80% of the traffic for the entire system. This is not surprising, given the fact that the screens in the top quartile were designed to support the core business process while those in the lower quartiles supported exception handling and administrative functions.

Next, we analyzed the change request synopsis to determine what screens had undergone change as part of the evolving business process. We then performed a regression analysis to see the impact of determine whether these variables were predictive of one another. Table 2 presents the results of this regression analysis.

Next, we analyzed the change request synopsis to determine what screens had undergone change as part of the evolving business process. We then correlated the number of change requests for individual screens with the screen utilization. We also correlated the type of screen (update, admin, or query) with the number of changes. Table 2 presents the results of correlating the change requests with the screen usage (average hits), the module from which the screen originated, the screen type and the number of change requests.

Correlations

		Change Requests	Average Hits	Module	Screen Type
Pearson Correlation	Change Requests	1.000	.534	-.019	-.008
	Average Hits	.534	1.000	.081	-.113
	Module	-.019	.081	1.000	-.210
	Screen Type	-.008	-.113	-.210	1.000
Sig. (1-tailed)	Change Requests	.	.000	.366	.440
	Average Hits	.000	.	.072	.021
	Module	.366	.072	.	.000
	Screen Type	.440	.021	.000	.
N	Change Requests	325	325	325	325
	Average Hits	325	325	325	325
	Module	325	325	325	325
	Screen Type	325	325	325	325

TABLE 2 – CORRELATION BETWEEN CHANGE REQUESTS AND USAGE

The results in this Table show a positive association (0.53 Pearson Product Moment) between the changes to a screen and the number of hits that it receives as part of the overall set of screens. This is somewhat lower than expected as the screens from module four, which comprise the majority of the system underwent the fewest screen specific changes overall whereas screens from modules two and three underwent more changes for less of the system traffic (see Table 3). When broken down by module, three of the four modules exhibit a high correlation between change requests and screen usage. Two of the Pearson Product Moment correlations are at 0.69 whilst the other is at 0.60, which suggests a moderate relationship between the two factors. Furthermore, the regression shows significant relationships between screen type and average hits, as well as between change requests and the average number of hits for the screen. Hence, the number of hits and change requests can be shown to be predictors of one another.

Description	Number	Percent
Module 1 Screens	50	15.38
Module 2 Screens	37	11.38
Module 3 Screens	100	30.77
Module 4 Screens	138	42.46
Total Screens	325	

TABLE 3 –SCREEN CHANGES BY ACCESS AMOUNT

As part of our investigation, we also looked at whether there was any relationship between the screen type (admin, update, or query) and the number of changes requested. All of these correlation values were within a range that suggested there was no relationship between the two variables.

We also examined the relationship between the average number of hits per screen and the number of changes the screen underwent. Table 4 shows the total screen changes and the number of changes for screens in the top 10%, top quartile, and the other three quartiles along with their percentage as part of the overall individual screen changes.

Description	Number	Percent
Total Screen Changes	277	
Top 10% Screen Changes	141	50.90
Top 25% Screens Changes	218	78.70
Other three quartiles	59	21.30

TABLE 4 –SCREEN CHANGES BY ACCESS AMOUNT

Here, we see a phenomenon similar to that displayed in Table 1. In particular, screens that were in the top 10% in terms of access received about 51% of the changes, and the top quartile received about 79% of the changes. The other three quartiles received only 21% of the changes. This suggests that more widely used screens experience more changes. Again, the fact that the most accessed screens form the core of the business process serves as a strong indicator of their continued transformation as the business process continues to evolve and adapt.

We also looked at the relationship between screen type, module, and the number of change requests. In particular, we conducted an analysis of variance on the number of change requests based on the module and the screen type as factors. This analysis (Table 5) shows that the screen type had a highly statistically significant impact on the number of change requests and further suggests that screens assist the core business process are more likely to undergo changes.

Tests of Between-Subjects Effects
Dependent Variable: Change Requests

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Corrected Model	186.538(a)	11	16.958	3.216	.000
Intercept	149.364	1	149.364	28.327	.000
Module	19.827	3	6.609	1.253	.290
Type	91.582	2	45.791	8.684	.000
Module * Type	63.290	6	10.548	2.001	.065
Error	1650.373	313	5.273		
Total	2073.000	325			
Corrected Total	1836.911	324			
a R Squared = .102 (Adjusted R Squared = .070)					

TABLE 5 –ANOVA FOR MODULE AND TYPE AS FACTORS FOR THE NUMBER OF CHANGE REQUESTS

Finally, we sought to examine information about system estimates and the time required to implement changes based on the priority assigned to those changes by the business process analysts. Here, we examined the summary information about the change requests and looked at the differences between the actual implementation date and the date of the estimated completion. Table 6 shows the average differences system estimates and actual implementation dates for the changes at each of the priority levels along with information on the standard deviation,

and variation. It also provides a 95% confidence interval for the mean.

Estimates based on Change Level	Avg	Std Dev	95% Alpha	Low 95% CI	High 95% CI	Variation
(1) ASAP	22.85	45.48	10.03	12.82	32.88	2068.44
(2) High	28.23	71.39	9.84	18.39	38.08	5096.01
(3) Medium	35.88	70.17	10.34	25.54	46.22	4268.76
(4) Low	54.22	97.02	39.65	14.57	93.87	9412.09
All	31.34	67.14	6.00	25.34	37.34	4515.29

TABLE 6 –SCREEN CHANGES BY ACCESS AMOUNT

Here, we see an average difference between the estimated date and the actual implementation date of approximately 31 days for all changes in the system. Some of the discrepancy between the estimated date and the actual implementation date can be attributed to the fact that changes were released in batches rather than on an individual basis.

There are two further items of note here. First, the higher priority changes had smaller average differences between the estimated date and the actual implementation date. These higher priority changes also had smaller standard deviations, variation, and confidence intervals. While all means overlapped at the 90%, 95%, and 99% confidence levels, the system estimates tended to be better for the higher priority changes.

We also examined the time required to implemented changes at various priority levels. Table 7 shows the average time to implement changes from each priority level along with information on the standard deviation, and variation. It also provides a 95% confidence interval for the mean.

Time to Implement	Avg	Std Dev	95% Alpha	Low 95% CI	High 95% CI	Variation
(1) ASAP	144.00	120.03	26.47	117.53	170.47	14406.69
(2) High	279.23	142.79	19.69	259.54	298.92	20040.26
(3) Medium	278.74	530.63	78.17	200.57	356.91	423587.82
(4) Low	159.87	137.37	56.14	103.73	216.01	18871.03

TABLE 7 –SCREEN CHANGES BY ACCESS AMOUNT

Here, we see that changes with the most urgent priority took the least amount of time to implement. Since these changes were deemed most necessary to support the business process, it stands to reason that they should be implemented in the most expeditious manner. Low priority changes also took a shorter amount of time to implement. This may be due to their lack of connection with the business process, which would indicate that they were less comprehensive in their nature. Changes in categories two and three took almost twice the time on average to implement as those in category one. They also had higher standard deviations, variances, and confidence intervals. This difference was significant at the 90% and 95% confidence levels, although only the difference between priority one and priority 2 changes was significant at the 99% confidence level.

V. CONCLUSION, DISCUSSION AND RECOMMENDATIONS

This paper provides several insights into system usage and change in the post implementation phase of the software development cycle. First, we observe that the portions of the system that define the core business process receive the vast majority of the system traffic. In fact, the top quartile of screens in terms of system usage received 96% of all system traffic. This means that it is entirely logical that such screens will undergo the majority of changes associated with the business process.

Our research has also yielded insight into the efficacy of using prioritization as part and parcel to the change control and configuration management process. To this end, we observed that changes with higher priority (those that support the business process) had a significantly shorter implementation time and a smaller difference between the estimated time and the actual implementation time.

The data contained in this research yields insight into the nature of changes within implemented information systems and serves as a quantitative verification of an important aspect of Leavitt's Model. Namely, changes in the business process drive changes in the information systems associated with that business process.

VI. FUTURE WORK

While this exploratory paper has provided several insights about the nature of system usage and changes in implemented information systems, there are several areas of future work.

First additional data sets from large scale systems that are used in industrial settings would be helpful to understand whether the findings in this paper extend to bespoke systems from other companies.

Second, we would like to examine the data in greater granularity to see if we more precisely determine any causality between the system changes and system utilization. In particular, we would like to examine monthly usage data to see if there are substantial increases to screens that experience changes.

Third, we know that the change requests examined in this document were originally proposed to augment the business process through required changes or through efficiency gain. We would like to correlate the change requests with the changes to the business process to understand the relationship between these two areas.

Finally, authors such as Sliwerski, et al. [28,29] and others examine the nature of bug fixes in open source products such as Apache and Mozilla. We would like to see if the bug discovery and fix patterns found in proprietary, bespoke information systems mirror those found in open source, end-user software.

ACKNOWLEDGMENTS

The author would like to acknowledge the help and support of Mr. Dennis O. Holiday and Mr. David Rogers who were instrumental in explaining the workings of the system and in providing the data for this paper and Mr. Jim Lawson who generously provided system usage data.

REFERENCES

[1] Bersoff, E. H., Henderson, V. D., and Siegel, S. G. 1978. Software Configuration Management. In Proceedings of the Software Quality Assurance Workshop on Functional and Performance Issues S. Jackson and J. A. Lockett, Eds., 9-17

[2] Bosch, J. 2004. Software Variability Management. In Proceedings of the 26th international Conference on Software Engineering (May 23 - 28, 2004). International Conference on Software Engineering. IEEE Computer Society, Washington, DC, 720-721.

[3] F. P. Brooks, Jr., The Mythical Man-Month: Essays on Software Engineering, Addison Wesley, 1995.

[4] Calabough, J. 1987. Software configuration—an NP-complete problem. In Proceedings of the Conference on the 1987 ACM SIGBDP-SIGCPR Conference (Coral Gables, Florida, United States, March 05 - 06, 1987). E. M. Awad, Ed. SIGCPR '87. ACM Press, New York, NY, 182-194.

[5] Chillakanti, P. 2004. Role-based information security: change management issues. In Proceedings of the 2004 international Symposium on information and Communication Technologies (Las Vegas, Nevada, June 16 - 18, 2004). ACM International Conference Proceeding Series, vol. 90. Trinity College Dublin, 134-139.

[6] Davis, A. M. and Bersoff, E. H. 1991. Impacts of life cycle models on software configuration management. Commun. ACM 34, 8 (Aug. 1991), 104-118.

[7] Dietel, K. 2004. Mastering IT change management step two: moving from ignorant anarchy to informed anarchy. In Proceedings of the 32nd Annual ACM SIGUCCS Conference on User Services (Baltimore, MD, USA, October 10 - 13, 2004). SIGUCCS '04. ACM Press, New York, NY, 188-190.

[8] Estublier, J., Leblang, D., Clemm, G., Conradi, R., van der Hoek, A., Tichy, W., and Wiborg-Weber, D. 2002. Impact of the research community for the field of software configuration management. In Proceedings of the 24th international Conference on Software Engineering (Orlando, Florida, May 19 - 25, 2002). ICSE '02. ACM Press, New York, NY, 643-644.

[9] Estublier, J., Leblang, D., Clemm, G., Conradi, R., Tichy, W., van der Hoek, A., and Wiborg-Weber, D. 2002. Impact of the research community on the field of software configuration management: summary of an impact project report. SIGSOFT Softw. Eng. Notes 27, 5 (Sep. 2002), 31-39.

[10] Estublier, J. 2000. Software configuration management: a roadmap. In Proceedings of the Conference on the Future of Software Engineering (Limerick, Ireland, June 04 - 11, 2000). ICSE '00. ACM Press, New York, NY, 279-289.

[11] Feiler, R, P. P. 1990. Software process support through software configuration management. In Proceedings of the 5th international Software Process Workshop on Experience with Software Process Models (Kennebunkport, Maine, United States, October 10 - 13, 1989). International Software Process Workshop. IEEE Computer Society Press, Los Alamitos, CA, 58-60.

[12] Gentleman, W. M., MacKay, A., and Stewart, D. A. 1989. Commercial realtime software needs different configuation management. In Proceedings of the 2nd international Workshop on Software Configuration Management (Princeton, New Jersey, United States, October 24 - 27, 1989). R. N. Taylor, Ed. ACM Press, New York, NY, 152-161

[13] German, D. M., Cubranić, D., and Storey, M. D. 2005. A framework for describing and understanding mining tools in software development. In Proceedings of the 2005 international Workshop on Mining Software Repositories (St. Louis, Missouri, May 17 - 17, 2005). MSR '05. ACM Press, New York, NY, 1-5

[14] Joeris, G. 1997. Change management needs integrated process and configuration management. In Proceedings of the 6th European Conference Held Jointly with the 5th ACM SIGSOFT international Symposium on Foundations of Software Engineering (Zurich, Switzerland, September 22 - 25, 1997). M. Jazayeri and H. Schauer, Eds. Foundations of Software Engineering. Springer-Verlag New York, New York, NY, 125-141.

[15] Koponen, T. and Hotti, V. 2005. Open source software maintenance process framework. In Proceedings of the Fifth Workshop on Open Source Software Engineering (St. Louis, Missouri, May 17 - 17, 2005). 5-WOSSE. ACM Press, New York, NY, 1-5.

[16] J. Laudon K. Laudon. Management Information Systems: New Approaches to Organizationand Technology. Prentice Hall Publishing, 1998.

[17] Leavitt, H. (1965) Applied Organizational Change in Industry, In: March, J. (ed.), Handbook of Organizations, Chicago: Rand McNally, 1144-1170.

[18] Lutfiyya, H. L., Marshall, A. D., Bauer, M. A., Martin, P., and Powley, W. 1997. Configuration maintenance for distributed applications management. In Proceedings of the 1997 Conference of the Centre For Advanced Studies on Collaborative Research (Toronto, Ontario, Canada, November 10 - 13, 1997). J. H. Johnson, Ed. IBM Centre for Advanced Studies Conference. IBM Press, 16.

[19] Mei, H., Zhang, L., and Yang, F. 2001. A software configuration management model for supporting component-based software development. SIGSOFT Softw. Eng. Notes 26, 2 (Mar. 2001), 53-58.

[20] Mockus, Audris, Fielding, Roy, Herbsleb, James. Two case studies of open source software development: Apache and Mozilla. ACM Trans. Softw. Eng. Methodol. 11, 3 (Jul. 2002), 309-346.

[21] Nguyen, T. N., Munson, E. V., and Thao, C. 2004. Fine-grained, structured configuration management for web projects. In Proceedings of the 13th international Conference on World Wide Web (New York, NY, USA, May 17 - 20, 2004). WWW '04. ACM Press, New York, NY, 433-442.

[22] Nguyen, T. N., Munson, E. V., Boyland, J. T., and Thao, C. 2005. An infrastructure for development of object-oriented, multi-level configuration management services. In Proceedings of the 27th international Conference on Software Engineering (St. Louis, MO, USA, May 15 - 21, 2005). ICSE '05. ACM Press, New York, NY, 215-224.

[23] Nguyen, T. N., Munson, E. V., and Boyland, J. T. 2004. Object-oriented, structural software configuration management. In Companion To the 19th Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications (Vancouver, BC, CANADA, October 24 - 28, 2004). OOPSLA '04. ACM Press, New York, NY, 35-36.

[24] Sato, S. and Panton, A. 2003. Using a change-management approach to promote customer-centered design. In Proceedings of the 2003 Conference on Designing For User Experiences (San Francisco, California, June 06 - 07, 2003). DUX '03. ACM Press, New York, NY, 1-11.

[25] Schuster, H., Neeb, J., and Schamburger, R. 1999. A configuration management approach for large workflow management systems. In Proceedings of the international Joint Conference on Work Activities Coordination and Collaboration (San Francisco, California, United States, February 22 - 25, 1999). D. Georgakopoulos, W. Prinz, and A. L. Wolf, Eds. WACC '99. ACM Press, New York, NY, 177-186.

[26] Shigo, O., Wada, Y., Terashima, Y., Iwamoto, K., and Nishimura, T. 1982. Configuration control for evolutional software products. In Proceedings of the 6th international Conference on Software Engineering (Tokyo, Japan, September 13 - 16, 1982). International Conference on Software Engineering. IEEE Computer Society Press, Los Alamitos, CA, 68-75.

[27] Simmonds, I. 1989. Configuration management in the PACT software engineering environment. In Proceedings of the 2nd international Workshop on Software Configuration Management (Princeton, New Jersey, United States, October 24 - 27, 1989). R. N. Taylor, Ed. ACM Press, New York, NY, 118-121

[28] Śliwerski, J., Zimmermann, T., and Zeller, A. 2005. When do changes induce fixes?. In Proceedings of the 2005 international Workshop on Mining Software Repositories (St. Louis, Missouri, May 17 - 17, 2005). MSR '05. ACM Press, New York, NY, 1-5.

[29] Śliwerski, J., Zimmermann, T., and Zeller, A. 2005. HATARI: raising risk awareness. In Proceedings of the 10th European Software Engineering Conference Held Jointly with 13th ACM SIGSOFT international Symposium on Foundations of Software Engineering (Lisbon, Portugal, September 05 - 09, 2005). ESEC/FSE-13. ACM Press, New York, NY, 107-110.

[30] Sommerville, I. 1996. Sixth international workshop on software configuration management. SIGSOFT Softw. Eng. Notes 21, 4 (Jul. 1996), 54-57. DOI= http://doi.acm.org/10.1145/232069.232083

[31] Thomas, I. 1989. Version and configuration management on a software engineering database. In Proceedings of the 2nd international Workshop on Software Configuration Management (Princeton, New Jersey, United States, October 24 - 27, 1989). R. N. Taylor, Ed. ACM Press, New York, NY, 23-25.

[32] Xizhe, J. 2001. Evaluation technique of software configuration management (poster session). In Proceedings of the 6th Annual Conference on innovation and Technology in Computer Science Education (Canterbury, United Kingdom). ITiCSE '01. ACM Press, New York, NY, 186.